MATH 7339 - Machine Learning and Statistical Learning Theory 2

Section. Complexity of models

1. **Degree of Freedom**
2. **Applications of DF**
3. **Vapnik–Chervonenkis (VC) Dimension**

Motivation

How to select a "good" model?

Best model is one expected to do well on future data, i.e. to generalize well.

With **sufficient** data: we can divide the data into:

- Full training set to form our final model. (Fit)
- Validation set, in which we can pre-test the model before we test it 'formally' on the set Test. (predict test quality)
- Test set to test the model, and compute its generalizability. (Report quality of final model)

Often the above full modeling protocol is not feasible (too few data).

Instead use predicted error estimates of an error prediction method based on the training data themselves.

Analytic predictors of test error from data within the training set:

- AIC [Akaike information criterion]
- BIC [Bayesian information criterion]
- MDL [minimal description length]
- VC [Vapnik-Chernovenkis dimension

Numerical predictors of test error:

- CV [cross-validation]
- Bootstrap

Motivation:

As we move between methods of estimating the underlying regression function for a learning problem, we want to compare estimates of test error between different methods/models.

We can use cross validation to split our training set into a training and test set, but in practice comparing cross validation curves between methods isn't straight forward.

For example, what does it mean to pick Ridge Regression with $\lambda = 150$ over the linear regression on three variables? Or K-nearest neighbors for K = 5? We would like some sort of measure of the relative complexity between estimators.

The notions of **degrees of freedom and VC-dimension** are often used to provide an abstraction of the numbers of "effective" parameters used to fit a model.

➢ Degrees of freedom (informal definition)

Roughly speaking, the **degrees of freedom (df) of a model** (fitting procedure) is the number of "**effective**" **free parameters** θ_i that are to be determined (i.e., to be fit) by a training set $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$.

Degrees of freedom represent the number of points of control of a system, model, or calculation. (The Effective Number of Parameters)

It provides a quantitative measure of estimator/model complexity.

Degrees of freedom = number of independent values – number of statistics

- In statistics, degrees of freedom is the number of free observations used to calculate a statistic.
- In machine learning, degrees of freedom is the free number of parameters of a model. (We will focus in this section.)

Review of OLS and Ridge Regressions:

Assume also that data $\mathcal{D} = (X, \vec{y})$ here are centered, i.e., mean is zero.

$$\hat{y} = \hat{h}_\theta(\vec{x}\,) = \vec{x}^T\vec{\theta} = \theta_1 x_1 + \cdots + \theta_d x_d$$

For Ridge regression,

$$\hat{\vec{\theta}}_{Ridge} = \underset{\vec{\theta}}{\operatorname{argmin}}\, J^{Ridge}(\vec{\theta}\,) = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

In particular, if $\lambda = 0$, then $\hat{\vec{\theta}}_{Ridge}$ is the OLS optimizer $\hat{\vec{\theta}}_{ols}$

Consider the singular value decomposition (SVD) of X

$$X = UDV^T$$

Here,

U is an $N \times d$ orthogonal matrix. $(U^T U = I_d)$

V is a $d \times d$ orthogonal matrix.

D is a $d \times d$ diagonal matrix, with diagonal entries σ_i, called the singular values of X.

Furthermore, the image spaces $\text{im}\, U = \text{im}\, X$.

$$X^T X = VDV^T$$

$$\vec{u}_i = \frac{X\vec{v}_i}{\|X\vec{v}_i\|} \text{ for } i = 1, \ldots, d.$$

Then have for ordinary least squares prediction vector

$$\vec{y}_{ols} = \text{Proj}_{\text{im(X)}}\vec{y} = H\vec{y} = X(X^TX)^{-1}X^T\vec{y}$$

Replace X by $X = UDV^T$

We have

$$\vec{y}_{ols} = \cdots = UU^T\vec{y} = \sum_{j=1}^{d} \vec{u}_j\vec{u}_j^T\vec{y}$$

Sum of projections onto orthonormal vectors $\vec{u}_j$.

Example: Ordinary Least Squares

Thus, the degrees of freedom above OLS regression is d since we are estimating $\theta_1, \ldots, \theta_d$.

But note we can also get the degrees of freedom from from taking the **sum of the coefficients** of the projections

$$\vec{y}_{ols} = H\vec{y} = UU^T\vec{y} = \sum_{j=1}^{d} \vec{u}_j \vec{u}_j^T \vec{y}$$

The degree of freedom df=d is also the dimension of the span of the columns of X, i.e., $\mathrm{df} = \dim(\mathrm{im}(X))$.

Finally, it is easy to show we also have here

$$\mathrm{df} = \mathrm{Tr}(H) = \mathrm{Tr}\left(\mathrm{UU^T}\right) = \mathrm{Tr}\,(\mathrm{UU^T}) = \mathrm{Tr}\,(\mathrm{I_d}) = d$$

In addition, $H = UU^T$

Now for ridge regression recall

$$\vec{y}_{Ridge} = X(X^T X + \lambda I)^{-1} X^T \vec{y} = H\vec{y}$$

Replace X by $X = UDV^T$

$$\vec{y}_{Ridge} = \cdots = UD(D^2 + \lambda I)^{-1} DU^T \vec{y}$$

$$= \sum_{j=1}^{d} \vec{u}_j \left[\frac{\sigma_j^2}{\sigma_j^2 + \lambda} \vec{u}_j^T \vec{y} \right] \equiv H_\lambda \vec{y}$$

Note that the **effective degrees of freedom** can now be "thought" of as

$$\mathrm{df} = \mathrm{Tr}(\mathrm{H}) = \mathrm{Tr}(\mathrm{H}_\lambda) = \sum_{j=1}^{d} \left[\frac{\sigma_j^2}{\sigma_j^2 + \lambda} \right] < d$$

That is the number of effective free parameters is smaller than before, i.e., we have 'less' degrees of freedom if we 'force' the θ_j to be small.

Note also that as $\lambda \to 0$, we have

$$\hat{\vec{\theta}}_{Ridge} \to \hat{\vec{\theta}}_{ols} \text{ and } df \to d$$

While as $\lambda \to \infty$, we have

$$\hat{\vec{\theta}}_{Ridge} \to 0 \text{ and } df \to 0$$

We can generally choose the best - for ridge regression by cross-validation (i.e. try different values and see which works best).

Effective degrees of freedom

With regard to the effective degrees of freedom (df), we can say that when $\lambda > 0$ above, we are still keeping all d predictors in the model -- there is no variable selection.

However, we are restricting the predictors $\widehat{\theta_i}$ by shrinking them, i.e., establishing defector constraints on them, equivalent to removing the freedom in choosing the θ_i.

Thus even though the degrees of freedom (number of parameters $\widehat{\theta_i}$) is still d, the effective degrees of freedom have been reduced below d when $\lambda > 0$.

➢ Degrees of freedom (Mathematical Definition)

Even though the concept it represents is quite broad, degrees of freedom has a rigorous mathematical definition.

Suppose the training data is generated by $y = h(\vec{x}) + \epsilon$, with $\mathrm{E}[\epsilon] = 0$, and $\mathrm{Var}(\epsilon) = \sigma^2$

Equivalent, $y^{(i)} = h\left(\vec{x}^{(i)}\right) + \epsilon_i$, for $i = 1, \ldots, N$.

Suppose ϵ_i are uncorrelated. (i.i.d. training data is a stronger assumption.)

Suppose we fit some model $\hat{y} = \hat{h}(\vec{x})$ to the training data.

Definition: The number of **degrees of freedom of $\hat{h}$** is

$$\mathrm{df}(\hat{h}) := \frac{1}{\sigma^2} \sum_{i=1}^{N} \mathrm{Cov}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Matrix Notation for Degrees of Freedom.

$$\mathrm{df}(\hat{h}) = \frac{1}{\sigma^2}\mathrm{Trace}(\mathrm{Cov}(\hat{\vec{y}}, \vec{y}))$$

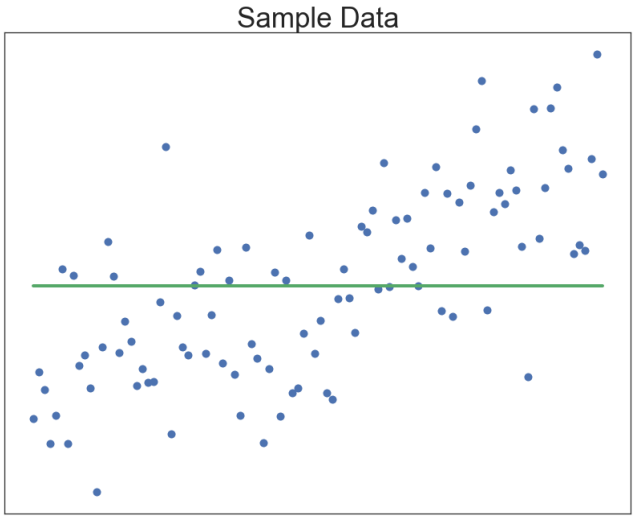
where

$$\hat{\vec{y}} = \begin{bmatrix} \hat{h}(\vec{x}^{(1)}) \\ \vdots \\ h(\vec{x}^{(N)}) \end{bmatrix}$$

This covariance treats only $y^{(i)}$ as random (but not $x^{(i)}$) for $i = 1, \ldots, N$

The definition of degrees of freedom looks at the amount of covariance between each point $y^{(i)}$ and its corresponding fitted values $\hat{y}^{(i)}$. We add these up over $i = 1, \ldots, n$, and divide the result by σ^2 (dividing by σ^2 gets rid of the dependence of the sum on the marginal error variance)

Example (mean predictor)



$$\hat{h}(x) = \theta_0 = \bar{y}$$

For **one** degree of freedom, we would expect a predictor with a single parameter, i.e., the constant predictor.

For example, we would expect the **mean predictor** $\hat{h}(x) = \bar{y}$

$$\bar{y} = \frac{1}{N}\left(y^{(1)} + \cdots + y^{(N)}\right)$$

to have a single degree of freedom. $\mathrm{df}(\bar{y}) = 1$.

Here, $y^{(i)} = h\left(\vec{x}^{(i)}\right) + \epsilon_i$ and recall that $\vec{x}^{(i)}$ is not treated as random variable.

Indeed, since ϵ_i are uncorrelated, the mean predictor

$$\hat{y} = \hat{h}(x) = \frac{1}{N}\left(y^{(1)} + \cdots + y^{(N)}\right)$$

has degrees of freedom:

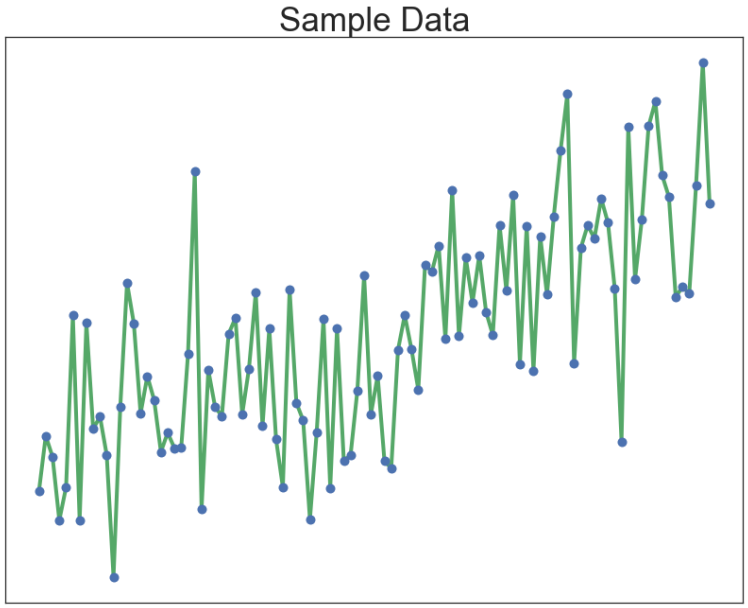
$$\mathrm{df}\left(\hat{h}\right) = \frac{1}{\sigma^2}\sum_{i=1}^{N} \mathrm{Cov}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

$$= \frac{1}{N\sigma^2}\sum_{i=1}^{N} \mathrm{Cov}\left(y^{(1)} + \cdots + y^{(N)}, y^{(i)}\right)$$

$$= \frac{1}{N\sigma^2}\sum_{i=1}^{N} \mathrm{Cov}\left(y^{(i)}, y^{(i)}\right) = 1$$

Recall that $\mathrm{Cov}\left(y^{(i)}, y^{(i)}\right) = \mathrm{Var}\left(y^{(i)}\right) = \sigma^2$

Example (identity predictors)



On the other hand, **an identity estimator** $\hat{y}^{(i)} = f\left(x^{(i)}\right) = y^{(i)}$ has $\boldsymbol{N}$ degrees of freedom:

$$\mathrm{df}\left(\hat{h}\right) = \frac{1}{\sigma^2}\sum_{i=1}^{N}\mathrm{Cov}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{\sigma^2}\sum_{i=1}^{N}\mathrm{Cov}\left(y^{(i)}, y^{(i)}\right) = N$$

Again, this intuitively makes sense: we would need at least N parameters to consistently make such a fit.

This notion of degrees of freedom gives a continuous measurement for the number of points we correctly guess, normalized by the number of standard deviations from the mean.

Example (OLS)

Assume also that data $\mathcal{D} = (X, \vec{y})$ here are centered. So, the data matrix X is an $N \times d$ matrix.

$$\hat{\vec{y}} = X\hat{\vec{\theta}} = X(X^T X)^{-1} X^T \vec{y}$$

Let us use the Trace (Tr) formula for the calculation:

$$\mathrm{df}(\hat{h}) = \frac{1}{\sigma^2} \mathrm{Tr}\left(\mathrm{Cov}(\hat{\vec{y}}, \vec{y})\right)$$

$$= \frac{1}{\sigma^2} \mathrm{Tr}\left(\mathrm{Cov}(X(X^T X)^{-1} X^T \vec{y}, \vec{y})\right)$$

$$= \frac{1}{\sigma^2} \mathrm{Tr}\left(X(X^T X)^{-1} X^T \mathrm{Cov}(\vec{y}, \vec{y})\right)$$

$$= \mathrm{Tr}\left(X(X^T X)^{-1} X^T\right)$$

Since Tr(AB)=Tr(BA),

$$= \mathrm{Tr}\left((X^T X)^{-1} X^T X\right) = Tr(I_d) = d$$

Proposition: Suppose a linear prediction has the form

$$\hat{\vec{y}} = \begin{bmatrix} \hat{h}\left(\vec{x}^{(1)}\right) \\ \vdots \\ h\left(\vec{x}^{(N)}\right) \end{bmatrix} = H\vec{y}, \text{ where } H \text{ is not depending on } \vec{y}.$$

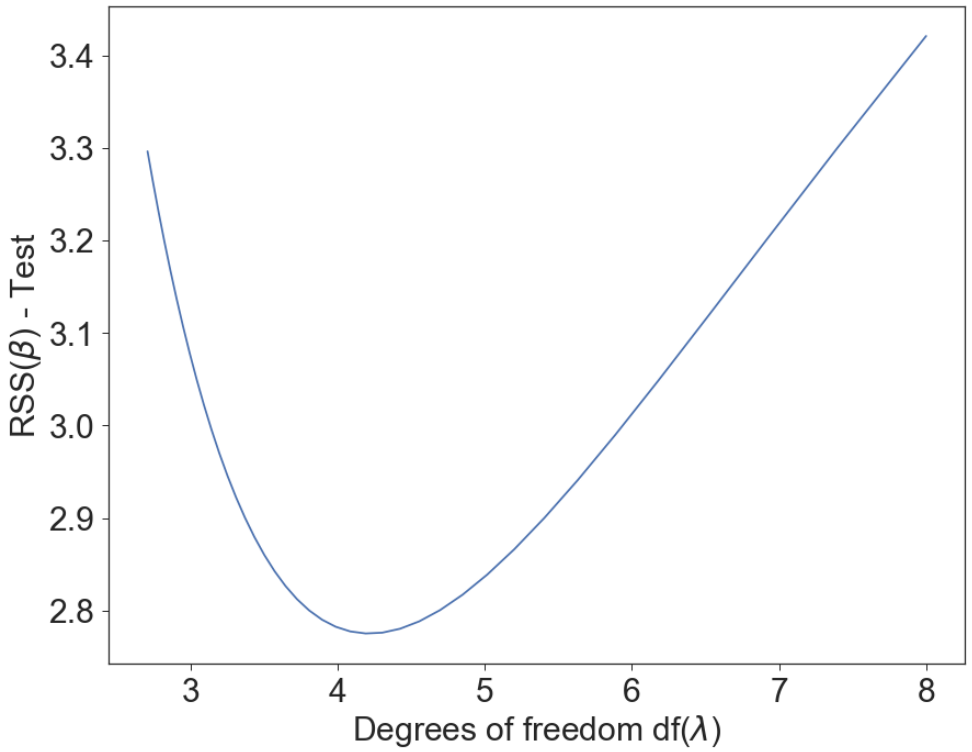
Then,

$$\mathrm{df}(\hat{h}) = \frac{1}{\sigma^2}\mathrm{T}r\left(\mathrm{Cov}(\hat{\vec{y}}, \vec{y})\right) = \frac{1}{\sigma^2}\mathrm{T}r\left(H\ \mathrm{Cov}(\vec{y}, \vec{y})\right) = Tr(H)$$

For example, in particular, this verifies that the ridge regression has degrees of freedom

$$\mathrm{df} = \sum_{j=1}^{d}\left[\frac{\sigma_j^2}{\sigma_j^2 + \lambda}\right] < d$$

Test error v.s. Degrees of freedom



This also allows us to properly chart the test error vs the degrees of freedom. Here we see the test error.

Example: Linear smoothers

Suppose a model (called linear smooth) has the form

$$\hat{f}(\vec{x}) = \sum_{j=1}^{N} w\left(\vec{x}, \vec{x}^{(j)}\right) \cdot y^{(j)}$$ for some function w(,)

Then, $$\hat{f}\left(\vec{x}^{i}\right) = \sum_{j=1}^{N} w\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right) \cdot y^{(j)}$$

Hence $$\hat{\vec{y}} = \begin{bmatrix} \hat{f}\left(\vec{x}^{(1)}\right) \\ \vdots \\ f\left(\vec{x}^{(N)}\right) \end{bmatrix} = W\vec{y}$$

where matrix W is defined as $W_{ij} = w\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$

Hence, $$\mathrm{df}\left(\hat{\vec{y}}\right) = \mathrm{Tr}(W) = \sum_{i=1}^{N} w\left(\vec{x}^{(i)}, \vec{x}^{(i)}\right)$$

Example. K-Nearest Neighbors (K-NN)

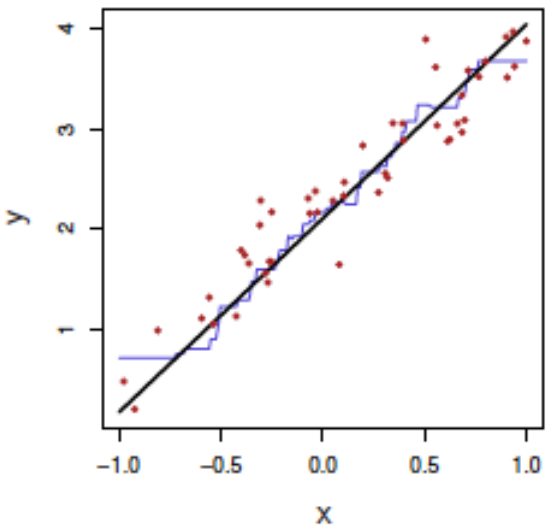
Consider k-nearest-neighbors regression with some fixed value of $K \geq 1$. Recall that here

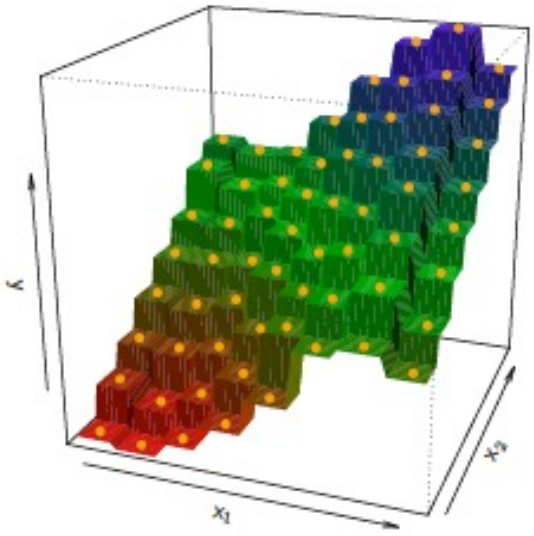
$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}^{(j)} \in \mathcal{N}_0} y^{(j)} = \sum_{j=1}^{N} w\left(\vec{x}, \vec{x}^{(j)}\right) \cdot y^{(j)}$$

where $$w\left(\vec{x}, \vec{x}^{(j)}\right) = \begin{cases} \frac{1}{k} & \text{if } \vec{x}^{(j)} \in \mathcal{N}_0 \\ 0 & \text{if } \vec{x}^{(j)} \notin \mathcal{N}_0 \end{cases}$$

So, $w\left(\vec{x}^{(i)}, \vec{x}^{(i)}\right) = \frac{1}{k}$

Hence $$\mathrm{df}\left(\hat{\vec{y}}\right) = \mathrm{Tr}(W) = \frac{\mathrm{N}}{\mathrm{k}}$$





Question: what happens for small k? Large k?

➢ Estimate degrees of freedom:

Above, we have some good examples that we can calculate the degrees of freedom by a closed formula. However, in some models, degrees of freedom can't always be calculated analytically, i.e., subset selection, Neural Network.

However, our definition $\mathrm{df}(\hat{h}) = \frac{1}{\sigma^2}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}^{(i)}, y^{(i)})$ is still well-defined.

We can estimate of the degrees of freedom, we can estimate the covariance terms $\mathrm{Cov}(\hat{y}^{(i)}, y^{(i)})$ by simulation.

A naive approach would be to use the **bootstrap** and calculate sample covariance.

$$\mathrm{SCov}\,(\vec{x}, \vec{z}) := \frac{\sum_{i=1}^{B}(x_i - \bar{x})(z_i - \bar{z})}{B-1}$$

For $b = 1$ to B (say $B = 1000$), we repeat the following steps:

1. Draw Bootstrap Samples $\left(\vec{x}_b^{(i)}, y_b^{(i)}\right)$, for $i = 1, \ldots, N$

2. Compute $\hat{\vec{y}}_b = \begin{bmatrix} \hat{h}\left(\vec{x}_b^{(1)}\right) \\ \vdots \\ \hat{h}\left(\vec{x}_b^{(N)}\right) \end{bmatrix}$ Denote $\vec{y}_b = \begin{bmatrix} y_b^{(1)} \\ \vdots \\ y_b^{(N)} \end{bmatrix}$

Then, we can estimate for degrees of freedom

$$\mathrm{df}(\hat{h}) \approx \frac{1}{B\sigma^2} \sum_{i=1}^{N} \sum_{b=1}^{B} \mathrm{SCov}\left(\hat{\vec{y}}_b, \vec{y}_b\right)$$

For simplicity, we assume that σ^2 is known; otherwise, we'd have to estimate it too. For example, in OLR linear model, the variance σ^2 is estimated by

$$\mathrm{s}^2 = \frac{RSS}{N-d-1} = \frac{\left\| X\hat{\vec{\theta}} - \vec{y} \right\|^2}{N-d-1} = \frac{\sum_{i=1}^{N} \left(\vec{x}^{(i)T} \hat{\vec{\theta}} - y^{(i)} \right)^2}{N-d-1}$$

Another better approach for estimating degrees of freedom is to use the *residual bootstrap*.

Training Error and Test Error

Training data $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right),\ i = 1, \ldots, N\right\} \subset \mathbb{R}^d \times \mathbb{R}$.

Model: prediction function: $f(\vec{x})$: $\mathbb{R}^d \to \mathbb{R}$

Loss Function: $L\left(y, f(\vec{x})\right)$ measuring error/distance of prediction $f(\vec{x})$.

Building a prediction model for we find which has the smallest loss.

Definition: The sample (training) error is average error of our model $\hat{f}(\vec{x})$ on its own training points $\mathcal{D}$:

$$err_{\mathcal{D}} = \frac{1}{N}\sum_{i=1}^{N} L\left(y^{(i)}, \hat{f}\left(\vec{x}^{(i)}\right)\right)$$

Definition: The (overall) **expected test error** is expected error on future data points $(\vec{X}, Y)$ based on the current (fixed) training set $\mathcal{D}$

$$Err_{\mathcal{D}} = E\left(L\left(Y, \hat{f}(\vec{X})\right)\right)$$

This is an expectation of true future error over all randomness.

If we also average over possible $\mathcal{D}$ the average error is just called Err.

Definition: Another predicted test error measure is the in-sample prediction error

$$Err_{in} = \frac{1}{N}\sum_{i=1}^{N} E_{Y_{new}}\left(L\left(Y_{new}^{(i)}, \hat{f}\left(\vec{x}^{(i)}\right)\right)\right)$$

Err_{in} = expected error based on same predicted function $\hat{f}\left(\vec{x}^{(i)}\right)$ from original training set $\mathcal{D}$ with same $\vec{x}^{(i)}$. **But** with **new** $Y_{new}^{(i)}$ from the same distributions.

Generally, $err_{\mathcal{D}}$ is too optimistic as predictor for the (true) $Err_{\mathcal{D}}$ or Err_{in}.

Next, we will

(1) Set up precise types of error measures (e.g. Errin vs. err)

(2) Review various error estimates and their relations to error analysis formulas mentioned above, e.g. G:, AIC, etc. as test error predictors from within the training set.

➢ Using degrees of freedom for Test error estimation

The difference between expected test and training errors, the **optimism** of the estimator $\hat{h}$, which directly related to Degrees of freedom.

Optimism$\equiv Err_{in} - err_{\mathcal{D}}$

$$= \frac{1}{N}\sum_{i=1}^{N} E_{Y_{new}}\left(L\left(Y_{new}^{(i)}, \hat{f}\left(\vec{x}^{(i)}\right)\right)\right) - \frac{1}{N}\sum_{i=1}^{N} L\left(y^{(i)}, \hat{f}\left(\vec{x}^{(i)}\right)\right)$$

This is still conditional on fixed data $\mathcal{D}$ and over the same training points $\vec{x}^{(i)}$.

Average over $y^{(i)}$ in $\mathcal{D}$, then we have the following result:

Theorem: $E_y(\textbf{Optimism}) = \frac{2}{N}\sum_{i=1}^{N} \mathrm{Cov}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2\sigma^2}{N}\mathrm{df}\left(\hat{f}\right)$

Hence:

$$E_y(Err_{in} - err_{\mathcal{D}}) = \frac{2\sigma^2}{N}\mathrm{df}(\hat{f})$$

In particular, for **Mean Squares Error,**

$$E\left[\frac{1}{N}\sum_{i=1}^{N}\left(y'^{(i)} - \hat{y}^{(i)}\right)^2\right] - E\left[\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2\right] = \frac{2\sigma^2}{N}\mathrm{df}(\hat{f})$$

Here the test points $y'^{(i)} = f\left(\vec{x}^{(i)}\right) + \epsilon^{(i)}$ with $E\left(\epsilon^{(i)}\right) = 0$ and $Var\left(\epsilon^{(i)}\right) = \sigma^2$.

Test Error Estimation

From above, a very natural **estimate** for the expected (Mean Square) test error is

$$T := \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \frac{2\sigma^2}{N}\,\mathrm{df}(\hat{\mathrm{h}})$$

Then, by the theorem of Optimism:

$$E(T) = E\left[\frac{1}{N}\sum_{i=1}^{N}\left(y'^{(i)} - \hat{y}^{(i)}\right)^2\right]$$

So, T is an **unbiased** estimate for the expected test error.

Hence if we knew an estimator's degrees of of freedom, then we could use T to approximate its test error.

note that this is a computationally efficient alternative to cross-validation (no extra computation really needed, beyond the training error)

➢ Model selection

Suppose our estimate $\hat{f} = f$ depends on a tuning parameter λ; also write $\hat{y}_\lambda$ for the fitted values at λ. Then over a grid of values, say $\lambda \in \{\lambda_1, \ldots, \lambda_m\}$, we compute the **Test error estimate**

$$T_\lambda := \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}_\lambda^{(i)}\right)^2 + \frac{2\sigma^2}{N}\mathrm{df}(\hat{f})$$

This may look familiar to you if we consider the case of **linear regression** on any number of predictor variables between 1 and d. Here, λ indexes the number of predictors used in a linear regression, and simply to make things look more familiar, we will rewrite this parameter as **k**. Hence $\mathrm{df}(f_k)$=k.

Hence $k \in \{1, \ldots, d\}$, and the above model selection criterion becomes

$$\hat{k} = \underset{k}{\mathrm{argmin}}\ \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}_k^{(i)}\right)^2 + \frac{2\sigma^2}{N}k$$

You may recall this as the C_p **information criterion** for model selection in **linear regression** (related to AIC, and then there's BIC, and RIC, ...)

Optimism is is **increasing** in k (dimension of model) and **decreasing** in N (size of training set)

In general:

"model complexity"

$\Leftrightarrow$ "too many parameters d fitting too small a size N of dataset $\mathcal{D}$"

$\Leftrightarrow$ "overfitting"

$\Leftrightarrow$ "optimism"

$\Leftrightarrow$ "unrealistically small error within training set $\mathcal{D}$"

Akaike information criterion (AIC) for log-likelihood models

Similarly, as C_p information criterion, the AIC is also the Test error **estimate.**

But more generally, we use AIC for log-likelihood models.

True predicted goodness of fit in test set =

$$-2E_Y(\ln P_\theta(Y)) = -\frac{2}{\mathrm{N}}\mathrm{E_Y}(\log \mathrm{lik}) + \frac{2\mathrm{d}}{\mathrm{N}}$$

Here, $\log lik = -\frac{2}{N}\sum_{i=1}^{N} \ln P_\theta(y^{(i)})$

$-2E_Y$(Log likelihood)$= -\frac{2}{\mathrm{N}}E_Y$(in training log likelihood)+ AIC difference

$$AIC := -\frac{2}{N}\mathbf{log\,lik} + \frac{2d}{N}$$

Classification Model: Y has some discrete probability distribution given $\vec{X}$.

Assume there is an unknown parameter vector $\vec{\theta}$ in the joint distribution of $(\vec{X}, Y)$

We want to compute or estimate distribution of Y given test point $\vec{X}$:

$$P_{\vec{\theta}}\left(Y \middle| \vec{X}\right) = \text{formula in } \vec{X}, Y \text{ and unknown parameter } \vec{\theta}.$$

Example: (logistics regression) $Y = 0,1$ with probability

$$P_{\vec{\theta}}\left(Y = 1 \middle| \vec{X}\right) = \frac{\mathrm{e}^{-\vec{\theta}^T \vec{X}}}{1 + \mathrm{e}^{-\vec{\theta}^T \vec{X}}}$$

$$P_{\vec{\theta}}\left(Y = 0 \middle| \vec{X}\right) = 1 - P_{\vec{\theta}}\left(Y = 1 \middle| \vec{X}\right) = \frac{1}{1 + \mathrm{e}^{-\vec{\theta}^T \vec{X}}}$$

If we use maximum likelihood to estimate $\hat{\vec{\theta}}$ (like in logistic regression) then (within training set $(\vec{x}^{(i)}, y^{(i)})$) the **cross-entropy loss function** to be minimized is

$$\log lik = -\frac{2}{N}\sum_{i=1}^{N} \ln P_{\theta}(y^{(i)})$$

Similar calculation to previous one gives:

$$AIC = -\frac{2}{N}\mathbf{log\ lik} + \frac{2d}{N}$$

log lik is the maximized log-likelihood

These C_p and AIC formulas give approximate unbiased estimators of Err_{in} for fixed (unchanging) "bases" (e.g., splines with fixed knots). We can't choose the bases adaptively in most cases (like in smoothing splines where knots are adaptively chosen from data).

This is because the following formula is only for basis fixed (e.g. fixed knots) beforehand.

$$\frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{2\sigma^2}{N}d$$

Linear and logistics.

There are additional methods:

- BIC (Bayes information criterion, this is the model with largest difference from **err**),
- MDL (minimal description length),
- DIC (deviance information criterion),
- VC (Vapnik-Chernovenkis) error bound formula.

➢ Vapnik–Chervonenkis (VC) Dimension

Consider dataset $\mathcal{D} = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right),\ i = 1, \ldots, N\right\}$ based on underlying probability distribution

$$\left(\vec{X}, Y\right) \sim P(\vec{x}, y)$$

We want to find estimate $\hat{f}(\vec{x})$ for regression function

$$f(\vec{x}) = E(Y|\vec{X} = \vec{x})$$

So far: we are trying to find estimates for $\boldsymbol{Err}_{\mathbf{in}} = E_Y\left(L\left(Y, \hat{f}\left(\vec{X}\right)\right)\right)$ (generalization error with same $\vec{x}^{(i)}$) in terms of $\boldsymbol{err}$ (training set error) by adding a correction term (e.g. $\frac{2\sigma^2}{N} d$)

We really want **estimate of full generalization error**, allowing for new overall training points i.e.,

$$Err = E_{Y,\vec{X}}\left(L\left(Y, \hat{f}\left(\vec{X}\right)\right)\right)$$

For **Least Squares Error,**

$$L\left(Y, \hat{f}\left(\vec{X}\right)\right) := \sum_{i=1}^{N} \left(y^{(i)} - \hat{f}\left(\vec{x}^{(i)}\right)\right)^2$$

VC theory has an **upper bound on Err** if we choose f from any class of functions (not just linear functions or functions in a reproducing kernel Hilbert space(RKHS)).

Consider a class $\mathcal{F}$ of functions $f(\vec{x})$ we allow ourselves to use for classification.

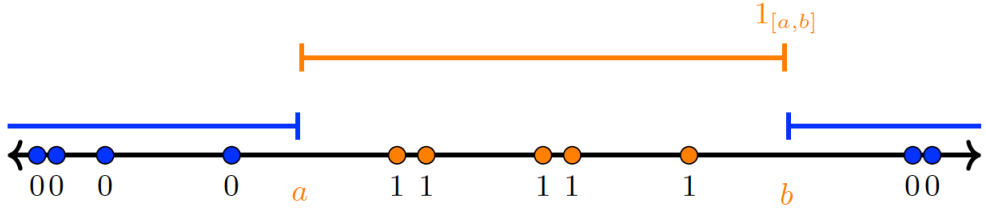
An inductive definition of VC dimension (indicator functions first)

We define VC dimension of $\mathcal{F}$ first for families $\mathcal{F}$ of indicator functions like

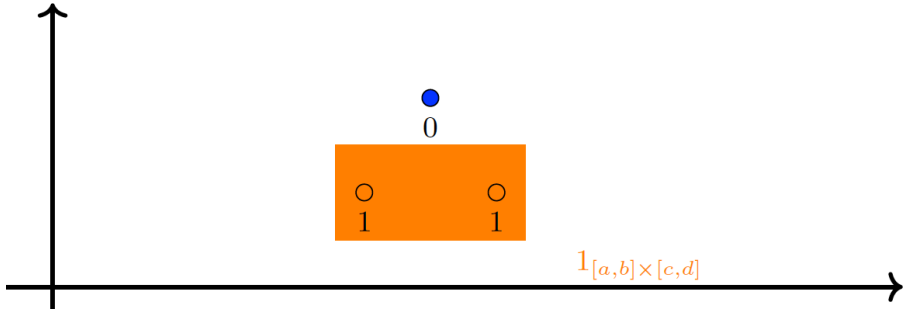
$$f(\vec{x}) = \mathbb{I}_A(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in A \\ 0 & \text{otherwise} \end{cases}$$

where, A is a subset of $\mathbb{R}^d$.

Example. (Indicator functions $\mathbb{I}_{[a,b]}(x)$ on $\mathbb{R}$, for any $a, b \in \mathbb{R}$)



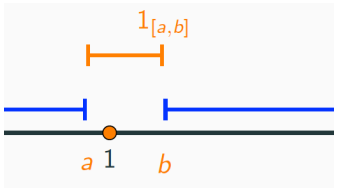
Example. (Indicator functions $\mathbb{I}_{[a,b]\times[c,d]}(\vec{x})$ on $\mathbb{R}^2$, for any $a, b, c, d \in \mathbb{R}$)

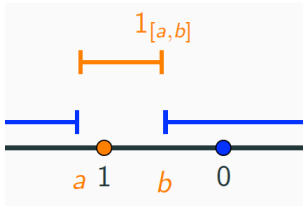


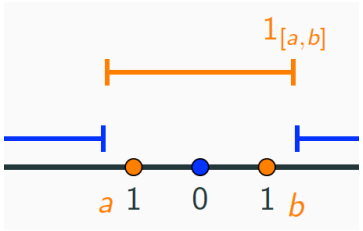
Definition: A family $\mathcal{F}$ of indicator functions $f(\vec{x}) = \mathbb{I}_A(\vec{x})$ on $\mathbb{R}^d$ **shatters** a **fixed** finite collection of points C if following holds:

For **every** subset $C_1 \subset C$, there is a function $\mathbb{I}_A(\vec{x})$ in the family that has value 1 on C_1 and 0 on $C - C_1$

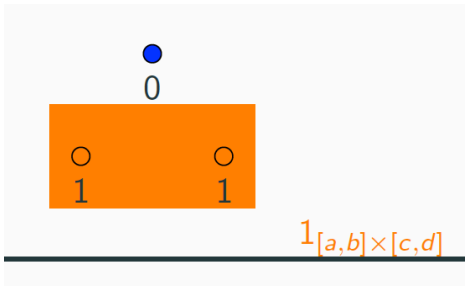
Example. (interval classifier: Indicator functions $\mathbb{I}_{[a,b]}(x)$ on $\mathbb{R}$, for any $a, b \in \mathbb{R}$)

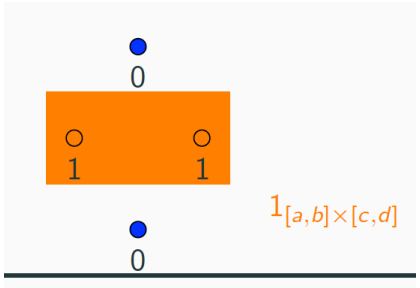






Example. (Indicator functions $\mathbb{I}_{[a,b]\times[c,d]}(\vec{x})$ on $\mathbb{R}^2$, for any $a, b, c, d \in \mathbb{R}$)





Definition: We say that this family $\mathcal{F}$ of indicator functions has VC dimension n if n is the **largest** number of points that $\mathcal{F}$ can always *shatter*.

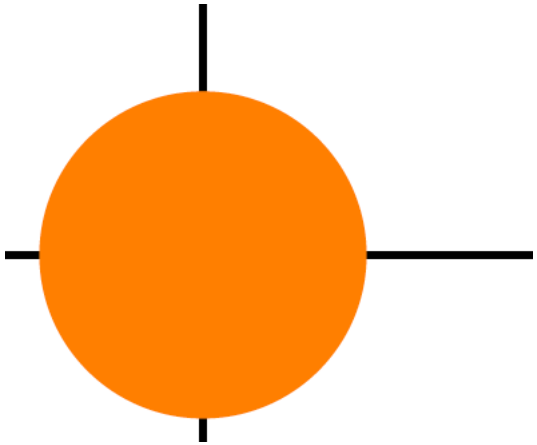
Note: It means that **at least** one set of size n that $\mathcal{F}$ can shatter

Example. (Indicator functions $\mathbb{I}_{[a,b]}(x)$ on $\mathbb{R}$)

The class $\mathcal{F}$ can clearly shatter a single point.
It can also shatter a two point set.
But an interval classifier cannot produce every labeling on the three-point set.
So, the VC-dimension is 2.

Example. (Indicator functions $\mathbb{I}_{[a,b]\times[c,d]}(\vec{x})$ on $\mathbb{R}^2$)

Question: What is the VC-dimension of the hypothesis class of indicator functions on the circles of radius $r \in \mathbb{R}^+$ centered at (0, 0)?



Question: How about indicator functions on the circles of radius $r \in \mathbb{R}^+$ centered at (a, b) for any $a, b \in \mathbb{R}$?

Question: How about 3-dimensional balls of radius $r \in \mathbb{R}^+$ centered at $\vec{0}$?

Then we can define VC dimension of more general classes of functions. Note that then we can do linear classification using functions like

$$\mathbb{I}_A(\vec{x}) = \mathbb{I}_{\{\vec{x} \mid \vec{\theta}^T \vec{x} \geq 0\}}(\vec{x})$$

Definition: A family $\mathcal{F}$ of general (non-indicator) functions $f(\vec{x})$ has VC dimension n if the collection of indicator functions $\{\mathbb{I}_{\{\vec{x} \mid f(\vec{x}) \geq a\}} \mid f \in \mathcal{F}, a \in \mathbb{R}\}$ has VC dimension n as defined above.

➢ Direct definition of VC Dimension

Definition: Suppose we have a class of functions $\{f(\vec{x}, \vec{\alpha})\}$ indexed by a parameter vector $\vec{\alpha}$, with $\vec{x} \in \mathbb{R}^d$.

The **VC dimension of the class** $\{\boldsymbol{f}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{\alpha}})\}$ is defined to be the largest number of points (in some configuration) that can be **shattered** by members of $\{f(\vec{x}, \vec{\alpha})\}$.

The VC dimension is another way of measuring the complexity of a class of functions by assessing how wiggly its members can be. For example, the VC dimension is a measure of complexity like $d =$ # features or # of basis functions earlier.

Estimate of full generalization error

We really want to use VC dimension to construct an estimate of (extra sample) prediction

$$Err = E_{Y,\vec{X}}\left(L\left(Y, \hat{f}\left(\vec{X}\right)\right)\right)$$

Theorem: If we fit N training points using a class of functions $\{f(x,\alpha)\}$ having VC dimension h, then with probability at least $1 - \eta$ over training. Denote

$$\epsilon = \frac{h\left[\ln\left(\frac{N}{h}\right) + 1\right] - \ln\frac{\eta}{4}}{N}$$

Then, for regression:

$$Err \leq \frac{err}{(1 - \sqrt{\epsilon})_+}$$

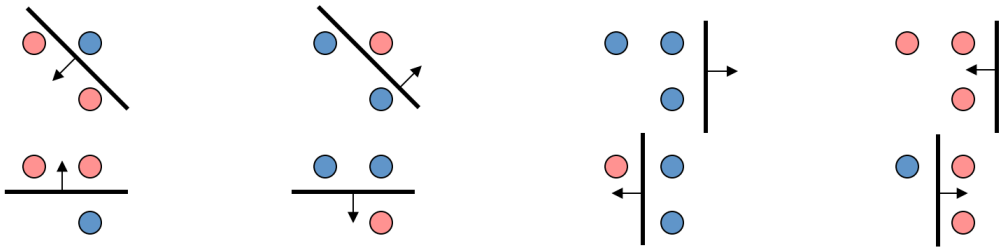
For binary classification, we have

$$Err \leq err + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot err}{\epsilon}}\right)$$

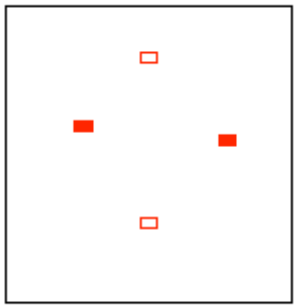
Example: Linear functions

Consider a class of linear functions $\mathcal{F} = \{f(\vec{x}, \vec{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2\}$ indexed by a parameter vector $\vec{\theta}$, with $\vec{x} \in \mathbb{R}^2$.

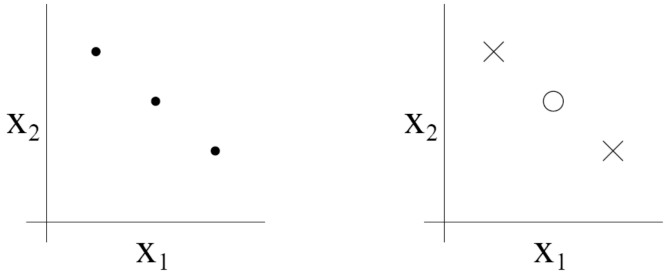
$$\mathbb{I}_A(\vec{x}) = \mathbb{I}_{\{\vec{x} \,|\theta_0+\theta_1 x_1+\theta_2 x_2 \geq 0\}}(\vec{x})$$



3 points shattered



4 points impossible



Even this C is not shattered by $\mathcal{F}$.

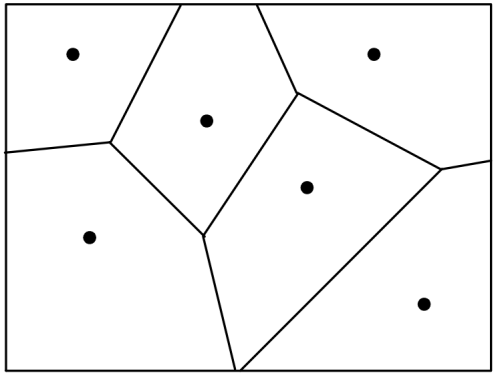
In general, the VC-dimension of the set of oriented hyperplanes in $\mathbb{R}^n$ is $n + 1$.

This is from the next linear algebra result:

Proposition: Consider some set of m points in $\mathbb{R}^n$. Choose any one of the points as origin. Then the m points can be shattered by oriented hyperplanes if and only if the position vectors of the remaining points are linearly independent.

Question: What is the VC-Dimension of K-nearest neighbors?

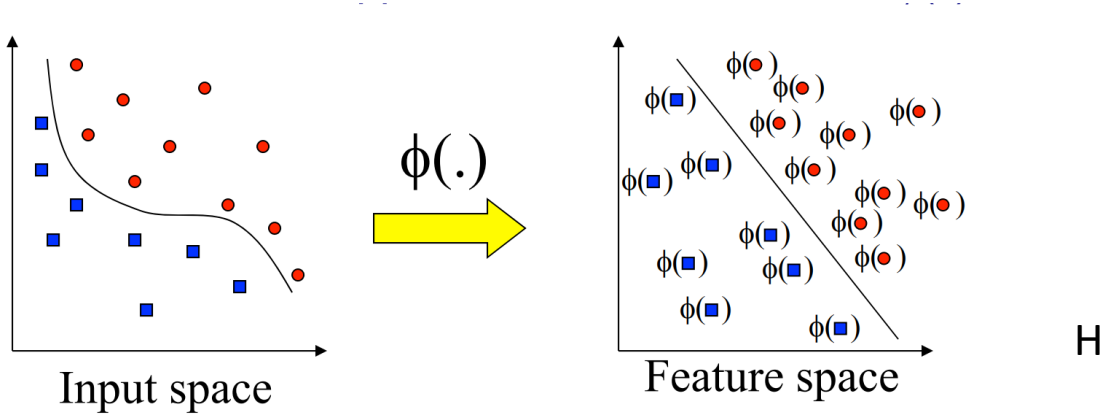
Decision Boundaries



The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

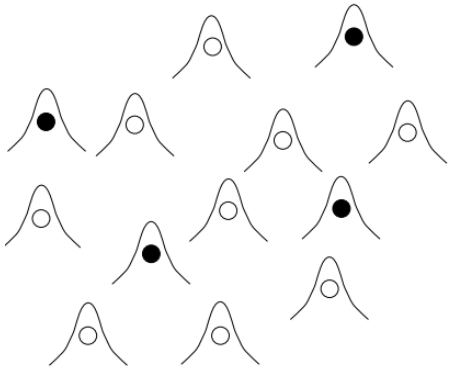
Infinite capacity does not guarantee poor performance

The VC Dimension of kernel SVMs

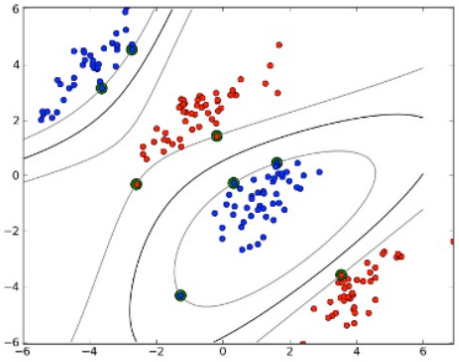


Theorem: Let K be a *kernel* which corresponds to a *minimal embedding space* H. Then the VC dimension of the corresponding support vector machine (where the error penalty C is allowed to take all values) is $\dim(\boldsymbol{H}) + \mathbf{1}$.

SVM with Gaussian Kernel has VC dimension ∞



[Figure from Chris Burges]



[Figure from mblondel.org]

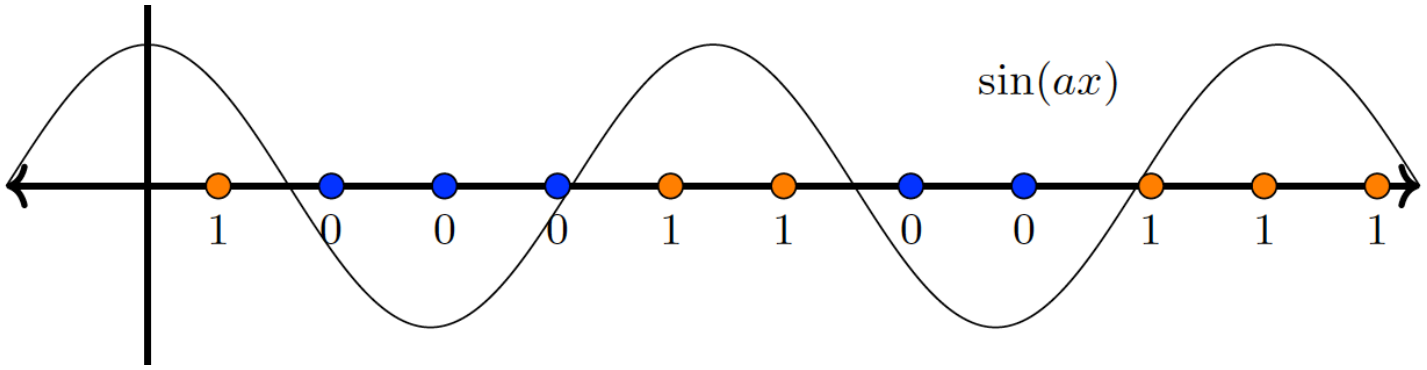
Question: Is it true that learning machines with many parameters would have high VC dimension, while learning machines with few parameters would have low VC dimension?

VC-Dimension and the number of parameters

VC dimension doesn't directly measure the number of parameters, but instead some “effective” number of parameters. For example, the classifier

$$f(x, a) \coloneqq \text{ceil}[\sin(ax)]$$

depends on only 1 parameter, but has infinite VC dimension.



References:

[Hastie, et.al.]. Sec 3.4, Chapter 7

•Degrees of Freedom in Deep Neural Networks, 2016. https://arxiv.org/abs/1603.09260

•Effective Degrees Of Freedom: A Flawed Metaphor, 2013. https://arxiv.org/abs/1312.7851